5

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/051,277	01/22/2002	Paul A. Lovvik	06502.0396-00000	7605

7590 05/01/2007

Finnegan, Henderson, Farabow,
Garrett & Dunner, L.L.P.
1300 I Street, N.W.
Washington, DC 20005-3315

EXAMINER	
PANNALA, SATHYANARAYAN R	

ART UNIT	PAPER NUMBER
2164	

MAIL DATE	DELIVERY MODE
05/01/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

MAILED

MAY 01 2007

Technology Center 2100

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application Number: 10/051,277
Filing Date: January 22, 2002
Appellant(s): LOVVIK ET AL.

John M. Mulcahy, Reg. No. 55,940
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed 1/5/2007 appealing from the Final Office Action mailed 9/5/2006.

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The examiner is not aware of any related appeals, interferences, or judicial proceedings, which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal. Examiner is relied upon the appellant's statement contained in the brief.

(3) Status of Claims

The statement of the status of the claims contained in the brief is correct.

(4) Status of Amendments After Final

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

(5) Summary of Claimed Subject Matter

The summary of invention contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct. Based on Appellant's argument, the rejection of claims under 35 U.S.C. § 101 is withdrawn.

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

Hendler et al. (USPA Pub 2002/0042833) hereinafter Hendler, and

Basin et al. (USPA Pub 2002/0120639) hereinafter Basin.

(9) Grounds of Rejection

The following grounds of rejection are applicable to the appealed claims:

Based on Appellant's argument, the rejection of claims under 35 U.S.C. § 101 is withdrawn.

Claims 1-19 are rejected under 35 U.S.C. § 103(a) which as being unpatentable over Hendler et al. (USPA Pub 2002/0042833) hereinafter Hendler and in view of Basin et al. (USPA Pub 2002/0120639) hereinafter Basin.

Claim Rejections - 35 USC § 103

1. The following is a quotation of 35 U.S.C. § 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

This application currently names joint inventors. In considering patentability of the claims under 35 U.S.C. 103(a), the examiner presumes that the subject matter of the various claims was commonly owned at the time any inventions covered therein were made absent any evidence to the contrary. Applicant is advised of the obligation under 37 CFR 1.56 to point out the inventor and invention dates of each claim that was not commonly owned at the time a later invention was made in order for the examiner to consider the applicability of 35 U.S.C. 103(c) and potential 35 U.S.C. 102(e), (f) or (g) prior art under 35 U.S.C. 103(a).

2. Claims 1-19 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Hendler et al. (USPA Pub 2002/0042833, date considered is 7/22/1998) hereinafter Hendler and in view of Basin et al. (USPA Pub 2002/0120639 date considered is 3/9/2000) hereinafter Basin.

3. As per independent claim 1, Hendler teaches a method and apparatus for streaming an archive files including JAVA archive file from a server to a client device (page 2, paragraph 0009). The client received stream file is stored at the client device in a JAVA Archive format (page 2, paragraph 0010). Hendler does not explicitly teach

receiving an un-extracted zip file. However, Basin teaches "un-extracted zip file" as these days, almost any file one downloads from the Internet is compressed (un-extracted) in some way. A standard compressed file or folder as it is sometimes called contains one or more files that were compressed into a single file or folder. (page 1, paragraph [0002]). Basin also teaches as an archive manager, which allows a user to open, view, modify and extract data from an existing archive or create a new archive (page 3, paragraph 0035).

Thus, it would have been obvious to one of ordinary skill in the data processing art at the time of the invention, have combined the teachings of the cited references because Basin's teachings would have allowed Hendler's method with an easy management and manipulation of archive files (Basin, page 1, paragraph [0008]). Hendler also teaches as Archive files could be of tar, zip files (page 7, paragraph 0067). Hendler teaches the claimed step of "receiving a stream of data containing a zip file, wherein the zip file comprises a set of files and a central directory" as Archive files can be streamed by extracting modules to client terminal 410 (Fig. 4, page 7, paragraph 0067).

Further, Hendler teaches the claimed step of "enabling a process to access contents of the central directory as the central directory is received" as the contents of each local file header is repeated in a central directory 640 located at the end of the ZIP archive (Fig. 6, page 7, paragraph 0069). Typically, the central directory information 640 will be streamed first, followed by the meta-information 631-632 and then the remaining modules 633-636 (Fig. 6, page 8, paragraph 0073)-74 and 0076).

4. As per dependent claim 2, Hendler teaches the claimed step of "the enabling step comprises providing an interface for accessing contents of the central directory" as the Zip file's central directory 640 includes an end of central directory record 640, the central directory is accessible as a part of the zip file (Fig. 6, page 7, paragraph 0070).

5. As per dependent claim 3, Hendler teaches the claimed step of "reading in a central directory file header" as the sizes of compressed files 601-606 can be obtained from the central directory header and the central directory header is read (Fig. 6, page 7, paragraph 0071). Further, Hendler teaches the claimed step of "providing an interface to access contents of the central directory file header" as the sizes of compressed files 601-606 can be obtained from the central directory file header and the central directory file header is accessed through an interface (Fig. 6, page 7, paragraph 0071).

6. As per dependent claim 4, Hendler teaches the claimed step of "reading in an end of central directory record" as the central directory 640 has an end record 647 and contains the total number of entries in the central directory end record and at the central directory end record is read to access the contents (Fig. 6, page 7, paragraph 0070). Further, Hendler teaches the claimed step of "providing an interface to access contents of the end of central directory record" as the central directory 640 has an end record

647 and contains the total number of entries in the central directory and the interface is provided to access the central directory end record (Fig. 6, page 7, paragraph 0070).

7. As per independent claim 5, Hendler and teaches a method and apparatus for streaming an archive files including JAVA archive file from a server to a client device (page 2, paragraph 0009). The client received stream file is stored at the client device in a JAVA Archive format (page 2, paragraph 0010). Hendler also teaches as Archive files could be of tar, zip files (page 7, paragraph 0067). Hendler teaches the claimed "a central processing unit" as the client computer 101 communicates with the server computer 102 (Fig. 1, page 3, paragraph 0029).

Further, Hendler teaches the claimed "an application program configured for execution by the central processing unit" as the server 401 provides a streaming-enabled version of application 500 to the client 410 (Fig. 4-5, page 6, paragraph 0060). Further, Hendler teaches the claimed "a receiving module, initiated by the application program, for receiving a streamed zip file" as the receiving device 410 includes a streaming executer 416 that controls the receipt of the streamed modules (Fig. 4, 6, page 8, paragraph 0075).

Hendler does not explicitly teach receiving an un-extracted zip file. However, Basin teaches "un-extracted zip file" as these days, almost any file one downloads from the Internet is compressed (un-extracted) in some way. A standard compressed file or folder as it is sometimes called contains one or more files that were compressed into a single file or folder (page 1, paragraph 0002). Basin also teaches as an archive manager, which allows a user to open, view, modify and extract data from an existing archive or create a new archive (page 3, paragraph 0035).

Thus, it would have been obvious to one of ordinary skill in the data processing art at the time of the invention, have combined the teachings of the cited references because Basin's teachings would have allowed Hendler's system with an easy management and manipulation of archive files (Basin, page 1, paragraph 0008).

Finally, Hendler teaches the claimed step of "an interface module, initiated by the application program, for accessing contents of a central directory of the streamed zip file as the central directory is received" as the integration of streamed modules with executing modules are provided by client 410 dynamic module linking facilities (Fig. 4, page 7, paragraph 0064-65 and Fig. 6, page 8, paragraph 0073-74 and 0076).

8. As per dependent claim 6, Hendler teaches the claimed "the interface module is a Java class comprising a central header subclass and a central directory subclass" as the streamed modules have a Java class comprising central directory headers 641-646 and central directory 640 as subclasses (Fig. 6, page 9, paragraph 0084).

9. As per dependent claim 7, Hendler teaches the claimed "the receiver module reads in a central directory header as an object instance of the central header subclass" as the central directory class headers 641-646 and the stream executor 416 controls the activity of the client 410 (Fig. 6, page 9, paragraph 0069-0070).

10. As per dependent claim 8, Hendler teaches the claimed "the central header subclass comprises a set of methods for accessing contents of the object instance" as the central directory headers as subclasses 641-646 and contents of object instances are accessed to get sizes and offset information (Fig. 6, page 8, paragraph 0074).

11. As per dependent claim 9, Hendler teaches the claimed "the receiver module reads in an end of central directory record as an object instance of the central directory subclass" as the central directory end record 647 and the stream executor 416 controls the activity of the client 410 (Fig. 6, page 7, paragraph 0070).

12. As per dependent claim 10, Hendler teaches the claimed "the central directory subclass comprises a set of methods for accessing contents of the object instance" as the central directory subclass and the stream executor 416 controls the activity of the client 410 including accessing objects 641-647 (Fig. 6, page 8, paragraph 0077).

13. As per independent claim 11, Hendler and teaches a method and apparatus for streaming an archive files including JAVA archive file from a server to a client device (page 2, paragraph 0009). The client received stream file is stored at the client device in a JAVA Archive format (page 2, paragraph 0010). Hendler also teaches as Archive files could be of tar, zip files (page 7, paragraph 0067). Hendler teaches the claimed "a receiving module for receiving a streamed an un-extracted zip file" as the receiving

device 410 includes a streaming executer 416 that controls the receipt of the streamed modules (Fig. 4, 6, page 8, paragraph 0075).

Hendler does not explicitly teach receiving an un-extracted zip file. However, Basin teaches "un-extracted zip file" as these days, almost any file one downloads from the Internet is compressed (un-extracted) in some way. A standard compressed file or folder as it is sometimes called contains one or more files that were compressed into a single file or folder. (page 1, paragraph 0002). Basin also teaches as an archive manager, which allows a user to open, view, modify and extract data from an existing archive or create a new archive (page 3, paragraph 0035).

Thus, it would have been obvious to one of ordinary skill in the data processing art at the time of the invention, to have combined the teachings of the cited references because Basin's teachings would have allowed Hendler's method with an easy management and manipulation of archive files (Basin, page 1, paragraph 0008).

Finally, Hendler teaches the claimed step of "interface module for accessing contents of a central directory of the streamed zip file as the central directory is received" as the integration of streamed modules with executing modules are provided by client 410 dynamic module-linking facilities (Fig. 4, page 7, paragraph 0064-65 and page 8, paragraph 0073-74 and 0076).

14. As per dependent claim 12, Hendler teaches the claimed "the interface module is a Java class comprising a central header subclass and a central directory subclass" as

the streamed modules have a Java class comprising central directory headers 641-646 and central directory 640 as subclasses (Fig. 6, page 9, paragraph 0084).

15. As per dependent claim 13, Hendler teaches the claimed "the receiver module reads in a central directory header as an object instance of the central header subclass" as the central directory class headers 641-646 and the stream executor 416 controls the activity of the client 410 (Fig. 4, 6, page 9, paragraph 0069-0070).

16. As per dependent claim 14, Hendler teaches the claimed "the central header subclass comprises a set of methods for accessing contents of the object instance" as the central directory headers as subclasses 641-646 and contents of object instances are accessed to get sizes and offset information (Fig. 6, page 8, paragraph 0074).

17. As per dependent claim 15, Hendler teaches the claimed "the receiver module reads in an end of central directory record as an object instance of the central directory subclass" as the end of central directory record 647 and the stream executor 416 controls the activity of the client 410 to read contents (Fig. 6, page 7, paragraph 0070).

18. As per dependent claim 16, Hendler teaches the claimed "the central directory subclass comprises a set of methods for accessing contents of the object instance" as the central directory subclass and the stream executor 416 controls the activity of the

client 410 including accessing accesses objects 641-47 (Fig. 4, 6, page 8, paragraph 0077).

19. As per independent claim 17, Hendler and teaches a method and apparatus for streaming an archive files including JAVA archive file from a server to a client device (page 2, paragraph 0009). The client received stream file is stored at the client device in a JAVA Archive format (page 2, paragraph 0010). Hendler also teaches as Archive files could be of tar, zip files (page 7, paragraph 0067). Hendler teaches the claimed "an interface stored in the memory, the interface for use with a receiver configured for receiving a streamed an zip file, wherein the zip file comprises a set of files and a central directory, the interface comprising a process for accessing contents of the central directory as the central directory is received" as the receiving device 410 includes a streaming executer 416 that controls the receipt of the streamed modules (Fig. 4, 6, page 8, paragraph 0073-76). Also, Hendler teaches as the integration of streamed modules with executing modules are provided by 410 dynamic module-linking facilities (Fig. 4, page 7, paragraph 0065).

Hendler does not explicitly teach receiving an un-extracted zip file. However, Basin teaches "un-extracted zip file" as these days, almost any file one downloads from the Internet is compressed (un-extracted) in some way. A standard compressed file or folder as it is sometimes called contains one or more files that were compressed into a single file or folder (page 1, paragraph 0002). Basin also teaches as an archive manager, which allows a user to open, view, modify and extract data from an existing archive or create a new archive (page 3, paragraph 0035).

Thus, it would have been obvious to one of ordinary skill in the data processing art at the time of the invention, have combined the teachings of the cited references because Basin's teachings would have allowed Hendler's method with an easy management and manipulation of archive files (Basin, page 1, paragraph 0008).

20. As per dependent claim 18, Hendler teaches the claimed "the process is a Java class comprising a set of methods for accessing contents of an object instance of the Java class, wherein the object instance comprises a central directory header read in by the receiver" as the central directory class headers 641-646 and the stream executor 416 controls the activity of the client 410 (Fig. 6, page 9, paragraph 0069-0070).

21. As per dependent claim 19, Hendler teaches the claimed "the process is a Java class comprising a set of methods for accessing contents of an object instance of the Java class, wherein the object instance comprises an end of central directory record read in by the receiver" as the end of central directory record 647 and the stream executor 416 controls the activity of the client 410 (Fig. 6, page 7, paragraph 0070).

(10) Response to Argument

A. Rejection of claims 1-19 under 35 U.S.C. § 101.

Based on Appellant's argument, the rejection of claims under 35 U.S.C. § 101 is withdrawn.

B. Rejection of independent claim 1 under 35 U.S.C. § 103(a).

Applicant argued as "The Examiner relies on Hendler et al. as teaching the claimed step of 'enable a process to access contents of the central directory as it is received,' as contents of each local file header is repeated in a central directory 640 located at the end of the zip archive [600]."

In response to the appellant's argument that Hendler does not teach, Examiner respectfully disagrees.

Claim 1: The appellant claims a method of accessing a streamed zip (i.e., an archive) files. The prior art used to reject this claim is Hendler et al. (USPA Pub 2002/0042833) hereinafter Hendler and Basin et al. (USPA Pub 2002/0120639) hereinafter Basin. Hendler teaches a method and apparatus for streaming an archive files including JAVA archive file from a server to a client device ("the invention features a method of streaming an archive file (such as a Java Archive file) from a server to a client device." page 2, paragraph 0009). The client received stream file is stored at the client device in a JAVA Archive format ("The received streamed files can be stored at the client device in a Java Archive format." page 2, paragraph 0010). Hendler also teaches that Archive files could be of tar, zip files ("Archive file such as CAB, tar, BINHEX, and ZIP files," page 7, paragraph 0067).

Hendler does not explicitly teach receiving an un-extracted zip file. For example, Microsoft Computer Dictionary defined the Zip file as a compressed file and PKZIP defined as a widely used shareware utility program for compressing files. Appellant is also depending on PKZIP ("One type of compressed file is a 'zip' file. A zip file is formatted according the zip file format provided by PKWare, Inc." see specification, page 1, paragraph 0005). Even though Zip files are understood as inherently compressed (un-extracted) files, Examiner specifically searched for more supporting prior art to teach explicitly compressed or un-extracted file and rejected under 35 U.S.C. 103(a). However, Basin teaches the claimed "un-extracted zip file" as these days, almost any file one downloads from the Internet is compressed (un-extracted) in some way. A standard compressed file or folder as it is sometimes called contains one or more files that were compressed into a single file or folder ("These days, almost any file one downloads from the Internet is compressed in some way." page 1, paragraph [0002]). Basin also teaches as an archive manager, which allows a user to open, view, modify and extract data from an existing archive or create a new archive ("an archive manager which allows a user to open, view, modify (add/delete), and extract data from an existing archive, or create a new archive." page 3, paragraph 0035).

Appellant's argument stated as "Moreover, Basin et al. teaches that 'generally, the contents of a compressed file cannot be accessed unless the archive is uncompressed.'

Basin et al. Paragraph 0004, lines 1-2." Basin stated this in the background section in order to claim that as per his invention, and teaches as, archive manager allows a user to open, view modify and extract data from an existing archive ("The present invention provides a software utility program that is integrated into Microsoft Windows Explorer for managing and manipulating archive files without leaving the Explorer environment." page 3, paragraph 0035).

The motivation used for combining as "Thus, it would have been obvious to one of an ordinary skill in the data processing art at the time of the invention, to have combined the teachings of the cited references because Basin's teachings would have allowed Hendler's method with an easy management and manipulation of archive files ("Accordingly, there is a need for a system and method for easy management and manipulation of archive files." Basin, page 1, paragraph 0008).

The claim first limitation is - receiving a stream of data containing a zip file, where the zip file comprises a set of files and a central directory. Hendler teaches this limitation as Archive files can be streamed by extracting modules to client terminal 410 ("main code module 501 may be received from the server 401 and begin execution at the client 410 while code libraries 510 and 515 are streamed from the server 401 to the client 410." Additionally, "Archive file such as CAB, tar, BINHEX, and ZIP files, including ZIP formatted Java Archive (JAR) files, can be streamed by extracting modules of data from the archive files and streaming those modules to a client terminal." and "A JAR (Java Archive) file is a data archive using the ZIP archive format to aggregate a collection of logically separate data files." further "Elements 611-616 of the archive 600 are known as local file headers. Each local file header precedes a file stored in the archive 600. The contents of each local file header is repeated (together with additional information) in a central directory 640 located at the end of the ZIP archive." Fig. 4, 6, page 7, paragraph 0067-69).

The claim other limitation is – enabling a process to access contents of the central directory as the central directory is received. Hendler teaches this limitation as the contents of each local file header is repeated in a central directory 640 located at the end of the ZIP archive ("Elements 611-616 of the archive 600 are known as local file headers. Each local file header precedes a file stored in the archive 600. The contents of each local file header is repeated (together with additional information) in a central directory 640 located at the end of the ZIP archive." Fig. 6, page 6, paragraph 0069. Further, he also teaches more details "Typically, the central directory information 640 will be streamed first, followed by the meta-information 631-632 (if present), and then the remaining modules 633-636." Para. - 73, "This size and offset information can be determined from the central directory file headers 641-647." Para – 74, "The receiving device 410 includes a streaming executor 416 that controls the receipt of the streamed modules 631-636, 640 and the storage of those modules in storage 411." Para. – 75. "To enable space allocation for the file 600, streaming of the modules 631-636, 640 may begin with streaming of the central directory module 640 (or, at least, of the end of central directory record 647)." Para. - 76, {page 8, paragraph 0073-76}).

Examiner used Basin's teaching as the second reference to explicitly teach **un-extracted files**. Basin teaches on page 1, paragraph 0002, as "these days, almost any file one downloads form the Internet is compressed in some way." A standard compressed file or folder is sometimes called contains one or more files that were compressed into a single file. Basin also teaches on page 3, paragraph 0035, as "the invention includes an archive manager, which allows a user to open, view, modify, and extract data form an existing archive, or create a new archive." **In fact, the specification does not support the word "un-extracted"**. As per the Appellant's specification, Interface module 218 accesses the central directory (see paragraph 047, line 3-4). Further, Hendler teaches that an executor can access the central directory contents to request the server if a module is not in the contents. ("The receiving device 410 includes a streaming executor 416 that controls the receipt of the streamed modules 631-636, 640 and the storage of those modules in storage 411. The component 416 can incrementally rebuild a JAR file at the receiving device 410 to store the received modules." Page 8, Paragraph 0075).

Hendler teaches as "the streamed files, central directory is accessed by the "Executor 416 determines whether procedure code 512-514 associated with the interrupt location has been received as part of the module stream 405 sent to client 410. If the code module has not yet been received, an explicit request may be sent from the client to the server" see Hendler, page 7, paragraph 0064. Further Hendler teaches as "Typically, the central directory information is 640 will be streamed first." see page 8, paragraph 0073. He also teaches as the receiving device 410 includes a streaming executor 416 that controls the receipt of the streamed modules. Therefore, from Hendler teaching, "executer will access the central directory and if not present the required module then order the server 401 for streaming of modules to the client" (see page 8, paragraph 0074).

Appellant has been misinterpreting the term "extracts" or "extracted" from the Hendler reference (see Appeal Brief, page 10, paragraph 1). For example, Examiner interpreted in a simple way as obtaining modules from the library at the server end, from Hendler teaching of "modules are extracted from contents of JAR (Java Archive) file at server end." (see page 8, paragraph [0074]).

C. The rejection of dependent claims 2-4 under 35 U.S.C. § 103(a).

Claims 2-4 are dependent on independent claim 1.

Claim 2: Claiming for an interface for accessing contents of the central file. Hendler teaches this claim as "executer 416 implements interface technology similar to that of a conventional run-time object code debugger" and "The central directory record 647 includes the total number of entries, the size of the central directory and the offset of the start of the central directory with respect to the start of an archive segment containing the central directory 640" (Fig. 4, 6, paragraph 0064 and 0070).

Claim 3: Claiming for reading a central directory file header and providing interface to access contents of the central directory file header. Hendler teaches as "the compressed size of the files 601-606 can be obtained from the central directory header" (Fig. 4, 6, page 7, paragraph 0064 and 0071).

Claim 4: Claiming for reading an end of central directory record and providing interface to access contents of the end of central record. Hendler teaches the claim as "central directory 640 has an end record 647 and contains the total number of entries in the central directory in addition to other information" (Fig. 4, 6, page 7, paragraph 0064 and 0069).

D. Rejection of independent claims 5 and 11 under 35 U.S.C. § 103(a).

Claims 5 and 11: The appellant claims a method of streaming an archive files. The prior art used to reject this claim is Hendler et al. (USPA Pub 2002/0042833) hereinafter Hendler and Basin et al. (USPA Pub 2002/0120639) hereinafter Basin. Hendler teaches a method and apparatus for streaming an archive files including JAVA archive file from a server to a client device ("the invention features a method of streaming an archive file (such as a Java Archive file) from a server to a client device." page 2, paragraph 0009). The client received stream file is stored at the client device in a JAVA Archive format ("The received streamed files can be stored at the client device in a Java Archive format." page 2, paragraph 0010page 2, paragraph 0010). Hendler also teaches as Archive files could be of tar, zip files ("Archive file such as CAB, tar, BINHEX, and ZIP files" page 7, paragraph 0067).

Hendler does not explicitly teach receiving an un-extracted zip file. For example, Microsoft Computer Dictionary defined the Zip file as a compressed file and PKZIP defined as a widely used shareware utility program for compressing files. Appellant also depending on PKZIP ("One type of compressed file is a 'zip' file. A zip file is formatted according the zip file format provided by PKWare, Inc." see specification at page 1, paragraph 0005). Even though Zip files are inherently compressed (un-extracted) files, Examiner specifically searched for more supporting prior art to teach explicitly compressed or un-extracted file and rejected under 35 U.S.C. 103(a). However, Basin teaches the claimed "un-extracted zip file" as these days, almost any file one downloads from the Internet is compressed (un-extracted) in some way. A standard compressed file or folder as it is sometimes called contains one or more files that were compressed into a single file or folder. ("These days, almost any file one downloads from the Internet is compressed in some way." page 1, paragraph [0002]). Basin also teaches as an archive manager, which allows a user to open, view, modify and extract data from an existing archive or create a new archive ("an archive manager which allows a user to open, view, modify (add/delete), and extract data from an existing archive, or create a new archive." page 3, paragraph 0035).

The motivation used for combining as "Thus, it would have been obvious to one of ordinary skill in the data processing art at the time of the invention, to have combined the teachings of the cited references because Basin's teachings would have allowed Hendler's method with an easy management and manipulation of archive files ("Accordingly, there is a need for a system and method for easy management and manipulation of archive files." Basin, page 1, paragraph 0008).

The claim first limitation is - a receiving module, initiated by the application program, for receiving a streamed zip file. Hendler teaches this limitation as Archive files can be streamed by extracting modules to client terminal 410 ("main code module

501 may be received from the server 401 and begin execution at the client 410 while code libraries 510 and 515 are streamed from the server 401 to the client 410." Additionally, "Archive file such as CAB, tar, BINHEX, and ZIP files, including ZIP formatted Java Archive (JAR) files, can be streamed by extracting modules of data from the archive files and streaming those modules to a client terminal." and "A JAR (Java Archive) file is a data archive using the ZIP archive format to aggregate a collection of logically separate data files." further "Elements 611-616 of the archive 600 are known as local file headers. Each local file header precedes a file stored in the archive 600. The contents of each local file header is repeated (together with additional information) in a central directory 640 located at the end of the ZIP archive." Fig. 4, 6, page 7, paragraph 0067-69).

Hendler does not explicitly teach receiving an un-extracted zip file. Examiner used Basin teaching as the second reference to teach **un-extracted files**. Basin teaches on page 1, paragraph 0002, as "these days, almost any file one downloads form the Internet is compressed in some way. A standard compressed file or folder is sometimes called contains one or more files that were compressed into a single file." Basin also teaches on page 3, paragraph 0035, as "the invention includes an archive manager, which allows a user to open, view, modify, and extract data form an existing archive, or create a new archive." **In fact, the specification does not support the word "un-extracted"**. The motivation used for combining as "Thus, it would have been obvious to one of ordinary skill in the data processing art at the time of the invention, to have combined the teachings of the cited references because Basin's teachings would have allowed Hendler's method with an easy management and manipulation of archive files ("Accordingly, there is a need for a system and method for easy management and manipulation of archive files." Basin, page 1, paragraph [0008]).

The claim other limitation is – an interface module, initiated by the application program, for accessing contents of a central directory of the streamed zip file as the central directory is received. Hendler teaches this limitation at Fig. 4, page 7, paragraph 0064-65, as executor 416 implements interface technology.

As per the Appellant's specification, Interface module 218 accesses the central directory (see specification at paragraph 047, line 3-4). Hendler teaches as the streamed files, central directory is accessed by the Executor 416 determines whether procedure code 512-514 associated with the interrupt location has been received as part of the module stream 405 sent to client 410. If the code module has not yet been received, an explicit request may be sent from the client to the server (see Hendler, page 7, paragraph 0064).

Further, Hendler teaches as Typically, the central directory information is 640 will be streamed first (page 8, paragraph 0073). He also teaches as the receiving device 410 includes a streaming executor 416 that controls the receipt of the streamed modules. Therefore, from Hendler teaching, executer will access the central directory and then order for streaming of modules to the client (see page 8, paragraph 0074).

Appellant argument stated as "Moreover, Basin et al. is not relied upon to teach, and does not teach, the claimed interface module." (see Appeal brief page 13, 2nd paragraph). Primary reference of Hendler teaches interface module at Fig. 4, page 7, paragraph 0064-65 as "Executor 416 implements interface technology similar to that of a conventional run-time object code debugger thereby allowing the executor 416 to intercept and process the interrupt generated by the application 500. The executor 416 intercepts the interrupt, determines an appropriate 4 Kilobyte code block that includes the interrupt statement. When an interrupt statement is executed by the application 500, replaces the determined code block with a received code module. If the appropriate code module has not yet been received, an explicit request may be sent from the client 410 to the server 401 to retrieve the code module prior to its insertion in the library 550 (Para – 64) and Integration of streamed modules with executing modules may be provided by client 410 dynamic module linking facilities (Para – 65)."

The secondary reference, Basin is not expected to teach the same limitation.

E. The rejection of dependent claims 6 and 12 under 35 U.S.C. § 103(a).

Clams 6 and 12: A system and computer program mechanism dependent claims on independent claims 5 and 11 respectively. Claiming for interface module is a Java class comprising a set of methods a central header subclass and a central directory subclass. Hendler teaches these claims at Fig. 6, page 9, paragraph 0084, as "the streamed modules have a Java class comprising central directory headers 641-646 and central directory 640 as subclasses. Further, Hendler teaches as the contents of each local file headers are repeated in a central directory 640" (Fig. 6, page 7, paragraph 0069).

F. Rejection of dependent claims 7-10 and 13-16 under 35 U.S.C. § 103(a).

Claims 7-10 depend from claim 6 and claims 13-16 depend on claim 12. Hendler teaches all these claims.

Claim 7: Appellant claiming for the receiver module reads in a central directory header as an object instance of the central header subclass. Hendler teaches as "the central directory class headers 641-646 and the stream executor 416 controls the activity of the client 410" (Fig. 6, page 9, paragraph 0069-0070).

Claim 8: Appellant is claiming for the central header subclass comprises a set of methods for accessing contents of the object instance. Hendler teaches as "the central directory headers as subclasses 641-646 and contents of object instances are accessed to get sizes and offset information" (Fig. 6, page 8, paragraph 0074).

Claim 9: Appellant is claiming for the receiver module reads in an end of central directory record as an object instance of the central directory subclass. Hendler teaches as "the central directory end record 647 and the stream executor 416 controls the activity of the client 410" (Fig. 6, page 7, paragraph 0070).

Claim 10: Appellant is claiming for the central directory subclass comprises a set of methods for accessing contents of the object instance. Hendler teaches as "the central directory subclass and the stream executor 416 controls the activity of the client 410 including accessing objects 641-647" (Fig. 6, page 8, paragraph 0077).

Claim 13: Appellant is claiming for the receiver module reads in a central directory header as an object instance of the central header subclass. Hendler teaches as "the central directory class headers 641-646 and the stream executor 416 controls the activity of the client 410" (Fig. 4, 6, page 9, paragraph 0069-0070).

Claim 14: Appellant is claiming for the central header subclass comprises a set of methods for accessing contents of the object instance. Hendler teaches as "the central directory headers as subclasses 641-646 and contents of object instances are accessed to get sizes and offset information" (Fig. 6, page 8, paragraph 0074).

Claim 15: Appellant is claiming for the receiver module reads in an end of central directory record as an object instance of the central directory subclass. Hendler teaches as "the end of central directory record 647 and the stream executor 416 controls the activity of the client 410 to read contents" (Fig. 6, page 7, paragraph 0070).

Claim 16: Appellant is claiming for the central directory subclass comprises a set of methods for accessing contents of the object instance. Hendler teaches as "the central directory subclass and the stream executor 0416 controls the activity of the client 410 including accessing accesses objects 641-47" (Fig. 4, 6, page 8, paragraph 0077).

G. Rejection of independent claim 17 under 35 U.S.C. § 103(a).

Claim 17: The appellant claims a memory for storing data for access by an application program being executed on a computer system. The preamble is very broad and general. The prior art used to reject this claim is Hendler et al. (USPA Pub 2002/0042833) hereinafter Hendler and Basin et al. (USPA Pub 2002/0120639) hereinafter Basin. Hendler teaches a method and apparatus for streaming an archive files including JAVA archive file from a server to a client device ("the invention features a method of streaming an archive file (such as a Java Archive file) from a server to a client device." page 2, paragraph 0009). The client received stream file is stored at the client device in a JAVA Archive format ("The received streamed files can be stored at the client device in a Java Archive format." page 2, paragraph 0010). Hendler also teaches as Archive files could be of tar, zip files ("Archive file such as CAB, tar, BINHEX, and ZIP files" page 7, paragraph 0067).

The claim limitation is – an interface stored in the memory, the interface for use with a receiver configured for receiving a streamed an un-extracted zip file, wherein the zip file comprises a set of files and a central directory, the interface comprising a process for accessing contents of the central directory as it is received. This limitation is also broad. Examiner assumes interface as an interface module, on the basis of the preamble. Hendler teaches as the receiving device 410 includes a streaming executer

416 that controls the receipt of the streamed modules ("Elements 611-616 of the archive 600 are known as local file headers. Each local file header precedes a file stored in the archive 600. The contents of each local file header is repeated (together with additional information) in a central directory 640 located at the end of the ZIP archive." Fig. 6, page 6, paragraph 0069. Further, he also teaches more details "Typically, the central directory information 640 will be streamed first, followed by the meta-information 631-632 (if present), and then the remaining modules 633-636." Para. - 73, "This size and offset information can be determined from the central directory file headers 641-647." Para – 74, "The receiving device 410 includes a streaming executor 416 that controls the receipt of the streamed modules 631-636, 640 and the storage of those modules in storage 411." Para. – 75. "To enable space allocation for the file 600, streaming of the modules 631-636, 640 may begin with streaming of the central directory module 640 (or, at least, of the end of central directory record 647)." Para. - 76, {page 8, paragraph 0073-76}). Also, Hendler teaches as the integration of streamed modules with executing modules are provided by 410 dynamic module-linking facilities (Fig. 4, page 7, paragraph 0065).

Hendler does not explicitly teach receiving an un-extracted zip file. For example, Microsoft Computer Dictionary defined the Zip file as a compressed file and PKZIP defined as a widely used shareware utility program for compressing files. Appellant is also depending on PKZIP ("One type of compressed file is a 'zip' file. A zip file is formatted according the zip file format provided by PKWare, Inc." see specification at page 1, paragraph 0005). Even though zip files are inherently compressed (un-extracted) files, Examiner specifically searched for more supporting prior art to teach explicitly compressed or un-extracted file and rejected under 35 U.S.C. 103(a). Basin teaches the claimed "un-extracted zip file" as "these days, almost any file one downloads from the Internet is compressed (un-extracted) in some way. A standard compressed file or folder as it is sometimes called contains one or more files that were compressed into a single file or folder." (page 1, paragraph 0002). Basin also teaches "as an archive manager, which allows a user to open, view, modify and extract data from an existing archive or create a new archive" (page 3, paragraph 0035).

The motivation used for combining as "Thus, it would have been obvious to one of ordinary skill in the data processing art at the time of the invention, to have combined the teachings of the cited references because Basin's teachings would have allowed Hendler's method with an easy management and manipulation of archive files ("Accordingly, there is a need for a system and method for easy management and manipulation of archive files." Basin, page 1, paragraph 0008).

H. Rejection of dependent claims 18 and 19 under 35 U.S.C. § 103(a).

Clams 18 and 19: The memory claims, both are dependent claims on independent claim 17. Appellant is claiming for process module is a Java class comprising a set of methods a central header subclass and a central directory subclass. Hendler teaches these claims at Fig. 6, page 9, paragraph 0084 as "the streamed modules have a Java class comprising central directory headers 641-646 and central directory 640 as subclasses." Further, Hendler teaches as "the contents of each local file headers are repeated in a central directory 640" (Fig. 6, page 7, paragraph 0069).

I. Conclusion

The references disclose the claimed invention of streaming of zip files. Hendler teaches streaming of Archive files in very details. In Fig. 6, Hendler shows the format of an example Java Archive (JAR) files. Hendler teaches that the typically the central directory is sent first and it contains each module header files, sizes and other information. He also teaches that modules could be sent with their heads or remove from them since the central directory has the same information. Hendler also teaches as an extracted (retrieved) file may be subdivided before streaming, for example, file 601 may a class file containing several Java classes. Even though Hendler teaches JAR files are streamed and inherent that they are in compressed form and for supporting the statement, the definition of PKZIP explained in the earlier paragraphs. The second reference by Basin has been introduced to explicitly teach un-extracted zip files. Examiner strongly feels that the number of file names in the central directory will be in thousands and not in millions and the time take to download will be less than a second.

For the above reasons, it is believed that the rejections should be sustained.

(11) Related Proceedings Appendix

No decision rendered by a court or the Board of Appeals and Interferences is identified by the examiner in the "Related Appeals and Interferences" section of this examiner's answer.

Respectfully Submitted,
[illegible]
Sathyanarayan Pannala
Primary Examiner

srp
April 25, 2007

Conferees:

1. Eddie Lee, TQAS/Appeals Specialist, TC 2100 [illegible]
EDDIE C. LEE
SUPERVISORY PATENT EXAMINER

2. Charles Rones, Supervisory Patent Examiner, Art Unit 2164

[illegible]

CHARLES RONES
SUPERVISORY PATENT EXAMINER